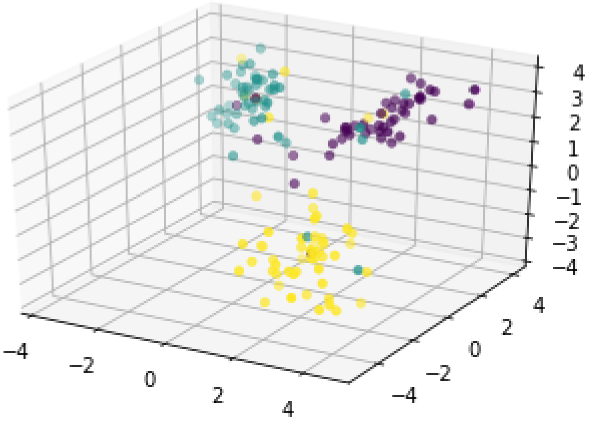
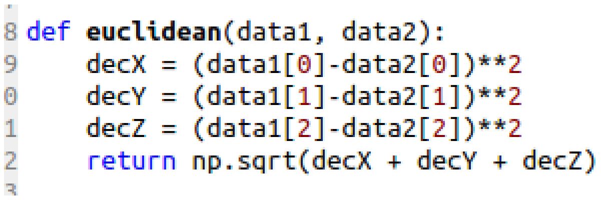
Helmi Satria (1301154325) - Probabilistic Neural Network

A. Yang harus Anda lakukan saat proses pembangunan model

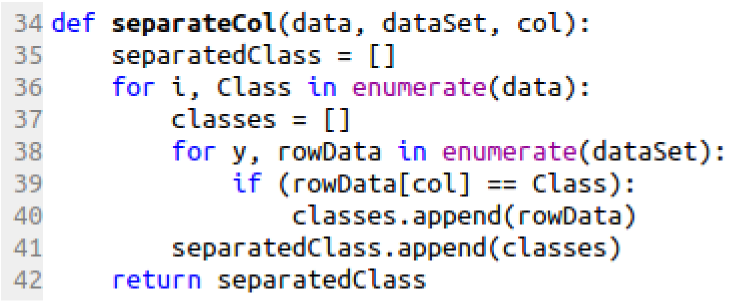
1. [10 POIN] Load data latih dari file yang diberikan, visualisasikan seluruh data menggunakan scatter plot.



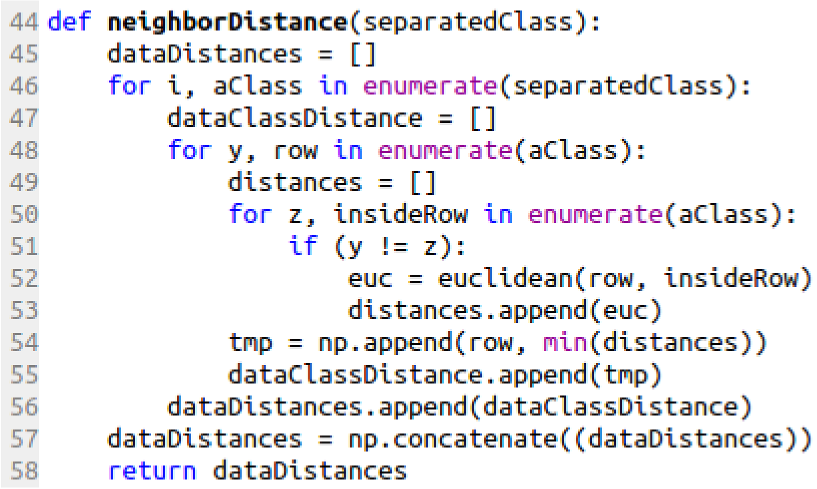
2. [25 POIN] Bangunlah fungsi-fungsi utama untuk mengklasifikasikan sebuah data menggunakan metode Jaringan Saraf Probabilistik.



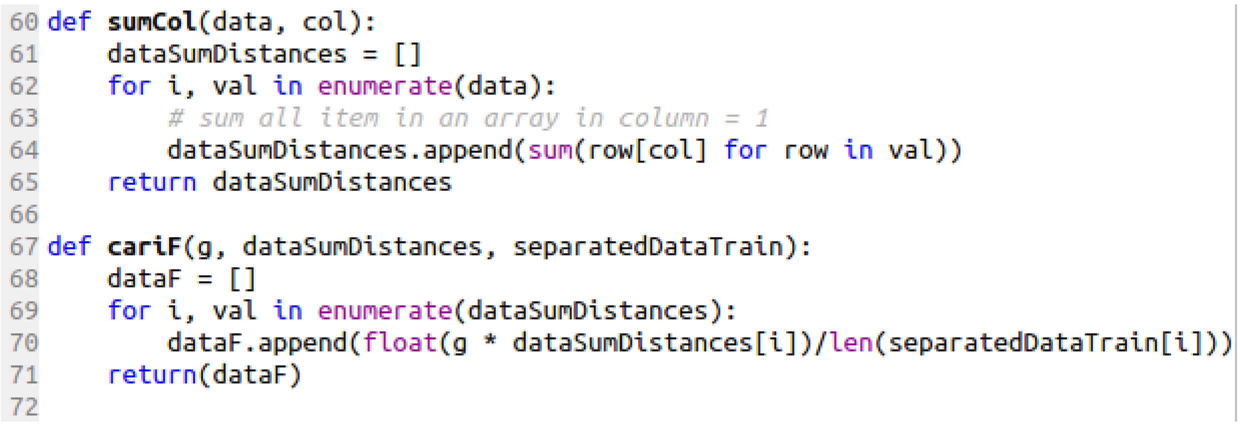
```python
def euclidean(data1, data2):
    decX = (data1[0]-data2[0])**2
    decY = (data1[1]-data2[1])**2
    decZ = (data1[2]-data2[2])**2
    return np.sqrt(decX + decY + decZ)
```



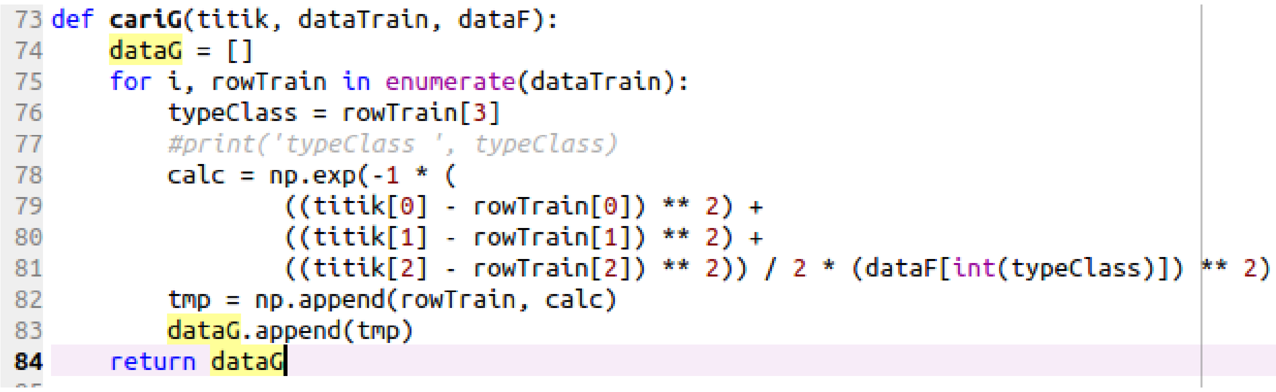
```python
def separateCol(data, dataSet, col):
    separatedClass = []
    for i, Class in enumerate(data):
        classes = []
        for y, rowData in enumerate(dataSet):
            if (rowData[col] == Class):
                classes.append(rowData)
        separatedClass.append(classes)
    return separatedClass
```



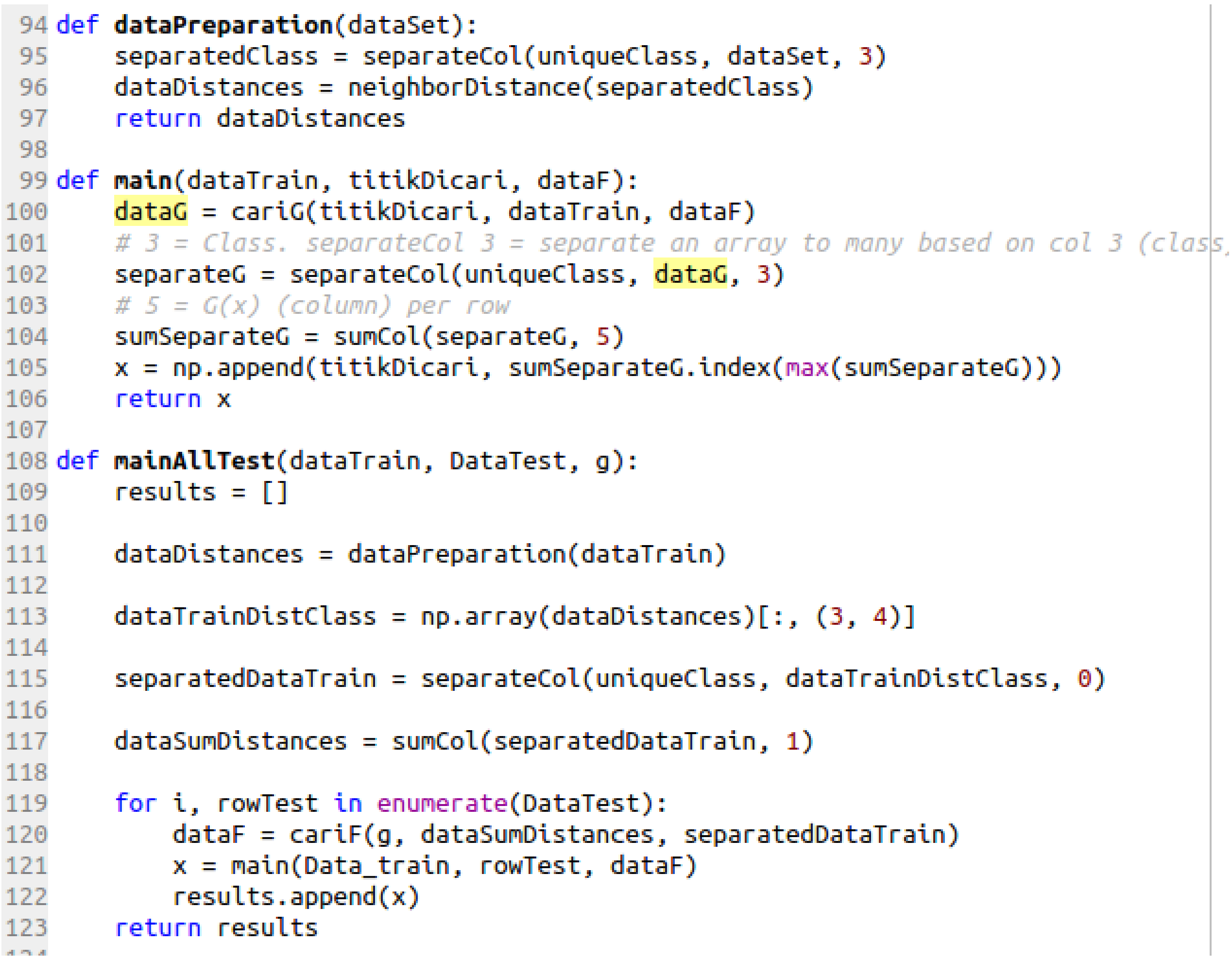
```python
def neighborDistance(separatedClass):
    dataDistances = []
    for i, aClass in enumerate(separatedClass):
        dataClassDistance = []
        for y, row in enumerate(aClass):
            distances = []
            for z, insideRow in enumerate(aClass):
                if (y != z):
                    euc = euclidean(row, insideRow)
                    distances.append(euc)
            tmp = np.append(row, min(distances))
            dataClassDistance.append(tmp)
        dataDistances.append(dataClassDistance)
    dataDistances = np.concatenate((dataDistances))
    return dataDistances
```



```python
def sumCol(data, col):
    dataSumDistances = []
    for i, val in enumerate(data):
        # sum all item in an array in column = 1
        dataSumDistances.append(sum(row[col] for row in val))
    return dataSumDistances

def cariF(g, dataSumDistances, separatedDataTrain):
    dataF = []
    for i, val in enumerate(dataSumDistances):
        dataF.append(float(g * dataSumDistances[i])/len(separatedDataTrain[i]))
    return(dataF)

```

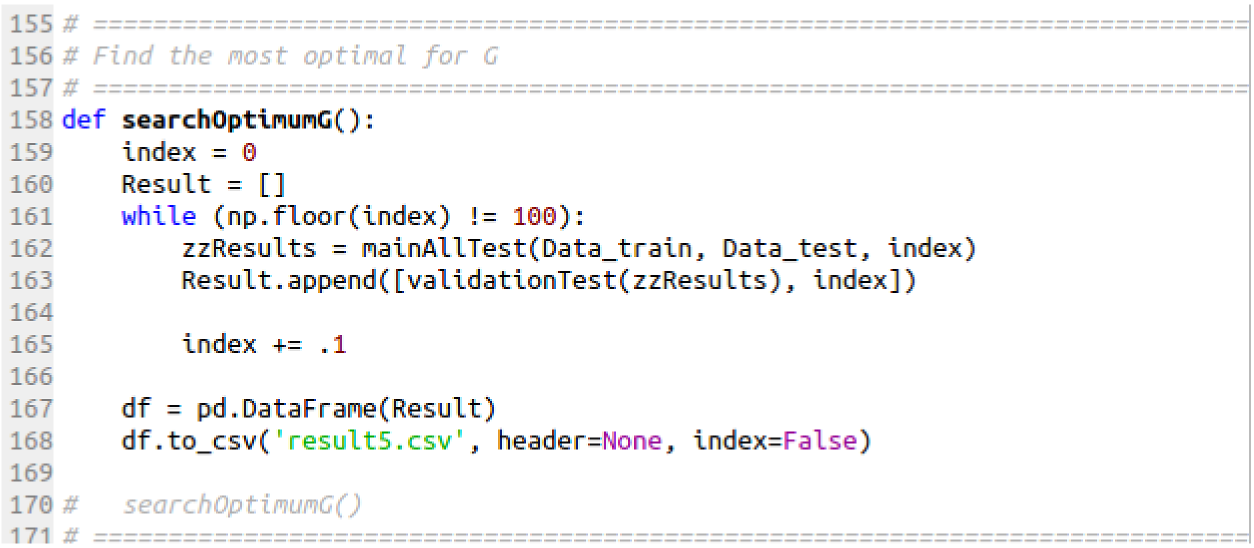


```python
def cariG(titik, dataTrain, dataF):
    dataG = []
    for i, rowTrain in enumerate(dataTrain):
        typeClass = rowTrain[3]
        #print('typeClass ', typeClass)
        calc = np.exp(-1 * (
                ((titik[0] - rowTrain[0]) ** 2) +
                ((titik[1] - rowTrain[1]) ** 2) +
                ((titik[2] - rowTrain[2]) ** 2)) / 2 * (dataF[int(typeClass)]) ** 2)
        tmp = np.append(rowTrain, calc)
        dataG.append(tmp)
    return dataG
```



```python
def dataPreparation(dataSet):
    separatedClass = separateCol(uniqueClass, dataSet, 3)
    dataDistances = neighborDistance(separatedClass)
    return dataDistances

def main(dataTrain, titikDicari, dataF):
    dataG = cariG(titikDicari, dataTrain, dataF)
    # 3 = Class. separateCol 3 = separate an array to many based on col 3 (class,
    separateG = separateCol(uniqueClass, dataG, 3)
    # 5 = G(x) (column) per row
    sumSeparateG = sumCol(separateG, 5)
    x = np.append(titikDicari, sumSeparateG.index(max(sumSeparateG)))
    return x

def mainAllTest(dataTrain, DataTest, g):
    results = []

    dataDistances = dataPreparation(dataTrain)

    dataTrainDistClass = np.array(dataDistances)[:, (3, 4)]

    separatedDataTrain = separateCol(uniqueClass, dataTrainDistClass, 0)

    dataSumDistances = sumCol(separatedDataTrain, 1)

    for i, rowTest in enumerate(DataTest):
        dataF = cariF(g, dataSumDistances, separatedDataTrain)
        x = main(Data_train, rowTest, dataF)
        results.append(x)
    return results
```

3. [30 POIN] Lakukan observasi untuk menentukan parameter-parameter terbaik yang akan digunakan di proses pengujian



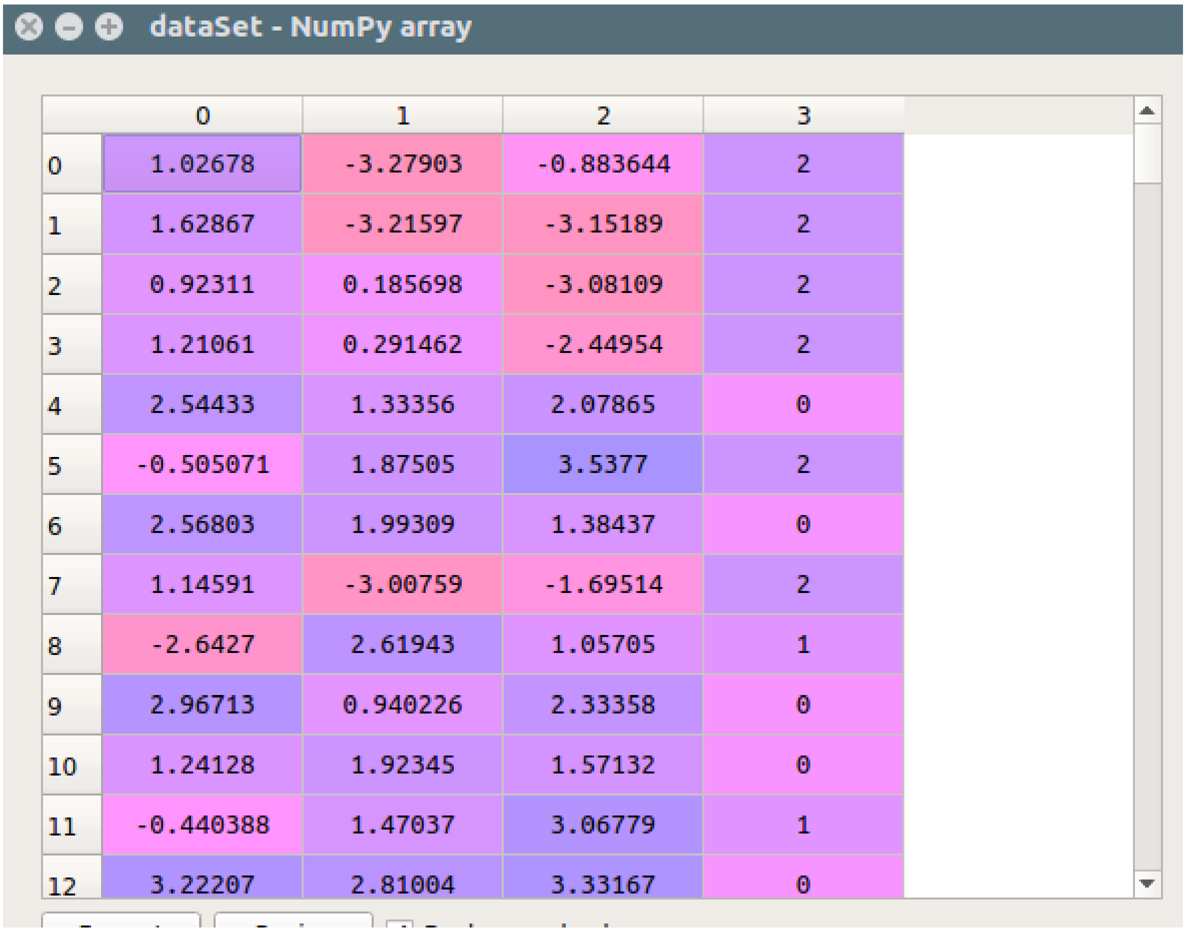
```python
# ==============================================================================
# Find the most optimal for G
# ==============================================================================
def searchOptimumG():
    index = 0
    Result = []
    while (np.floor(index) != 100):
        zzResults = mainAllTest(Data_train, Data_test, index)
        Result.append([validationTest(zzResults), index])

        index += .1

    df = pd.DataFrame(Result)
    df.to_csv('result5.csv', header=None, index=False)

#   searchOptimumG()
# ==============================================================================
```

B. [25 POIN] Sistem pengujian

1. [25 POIN] Sistem pengujian.

a. Load data latih dan data uji dari file yang diberikan.

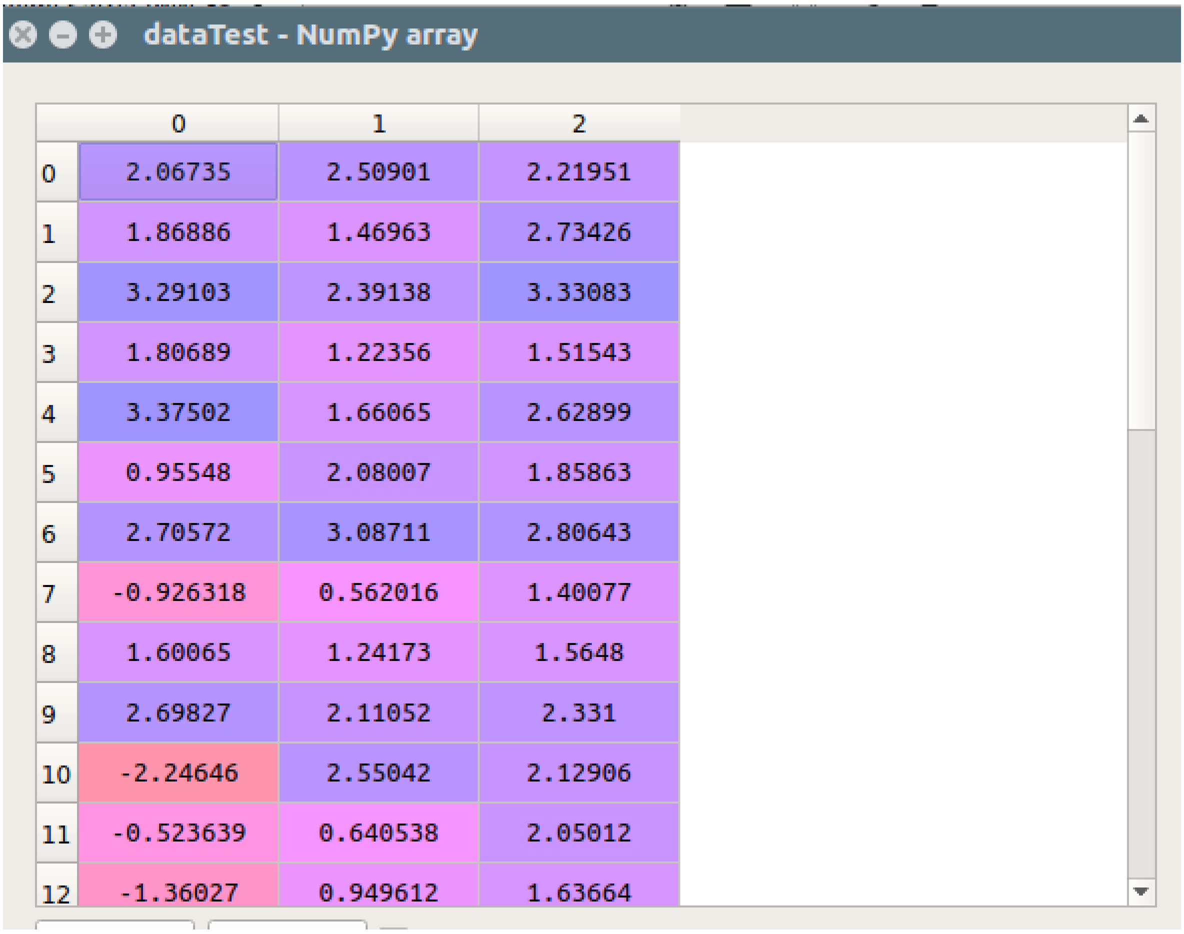
Data Train



dataSet - NumPy array

	0	1	2	3
0	1.02678	-3.27903	-0.883644	2
1	1.62867	-3.21597	-3.15189	2
2	0.92311	0.185698	-3.08109	2
3	1.21061	0.291462	-2.44954	2
4	2.54433	1.33356	2.07865	0
5	-0.505071	1.87505	3.5377	2
6	2.56803	1.99309	1.38437	0
7	1.14591	-3.00759	-1.69514	2
8	-2.6427	2.61943	1.05705	1
9	2.96713	0.940226	2.33358	0
10	1.24128	1.92345	1.57132	0
11	-0.440388	1.47037	3.06779	1
12	3.22207	2.81004	3.33167	0

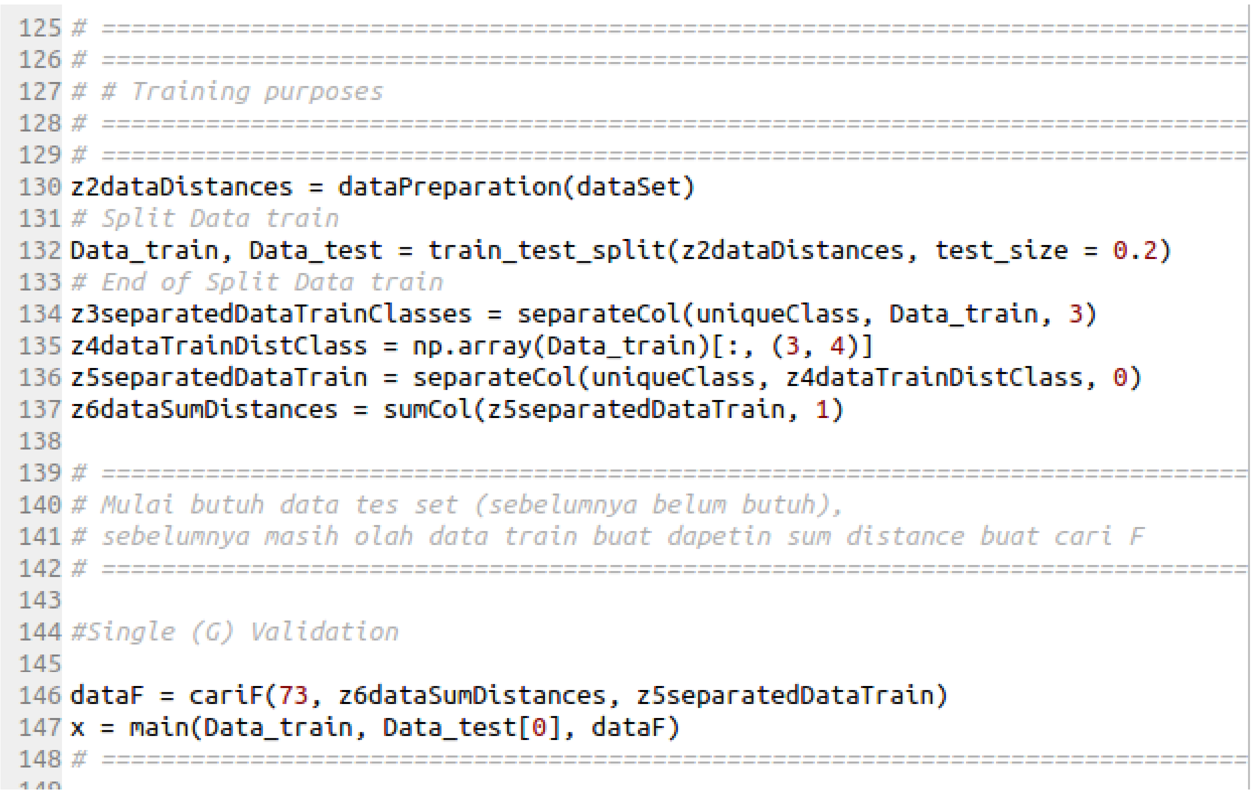
Data Test



dataTest - NumPy array

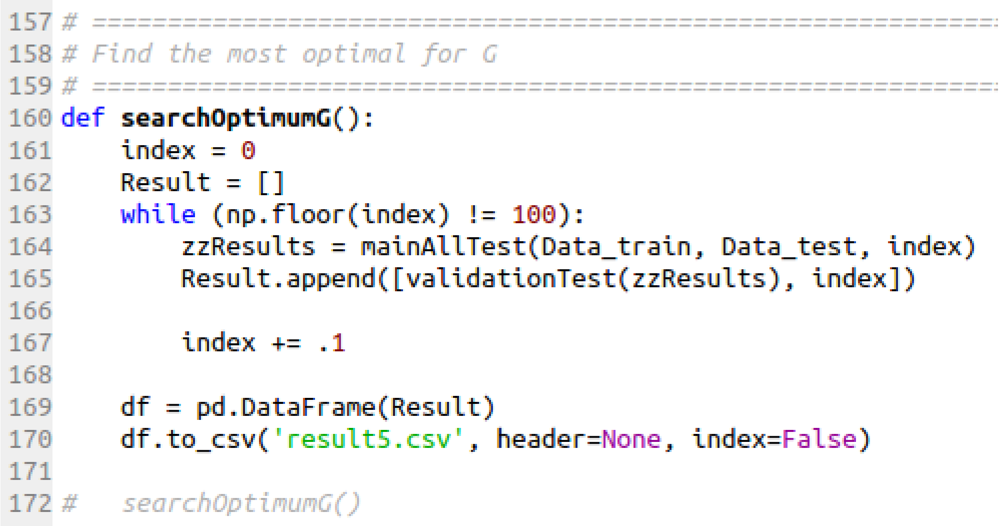
	0	1	2
0	2.06735	2.50901	2.21951
1	1.86886	1.46963	2.73426
2	3.29103	2.39138	3.33083
3	1.80689	1.22356	1.51543
4	3.37502	1.66065	2.62899
5	0.95548	2.08007	1.85863
6	2.70572	3.08711	2.80643
7	-0.926318	0.562016	1.40077
8	1.60065	1.24173	1.5648
9	2.69827	2.11052	2.331
10	-2.24646	2.55042	2.12906
11	-0.523639	0.640538	2.05012
12	-1.36027	0.949612	1.63664

b. Lakukan proses klasifikasi terhadap data uji menggunakan metode Jaringan Saraf Probabilistik dengan parameter yang sudah Anda tentukan saat proses observasi

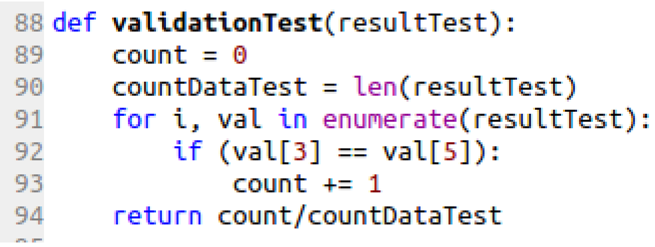


```python
# ============================================================================
# ============================================================================
# # Training purposes
# ============================================================================
# ============================================================================
z2dataDistances = dataPreparation(dataSet)
# Split Data train
Data_train, Data_test = train_test_split(z2dataDistances, test_size = 0.2)
# End of Split Data train
z3separatedDataTrainClasses = separateCol(uniqueClass, Data_train, 3)
z4dataTrainDistClass = np.array(Data_train)[:, (3, 4)]
z5separatedDataTrain = separateCol(uniqueClass, z4dataTrainDistClass, 0)
z6dataSumDistances = sumCol(z5separatedDataTrain, 1)

# ============================================================================
# Mulai butuh data tes set (sebelumnya belum butuh),
# sebelumnya masih olah data train buat dapetin sum distance buat cari F
# ============================================================================

#Single (G) Validation

dataF = cariF(73, z6dataSumDistances, z5separatedDataTrain)
x = main(Data_train, Data_test[0], dataF)
# ============================================================================
```

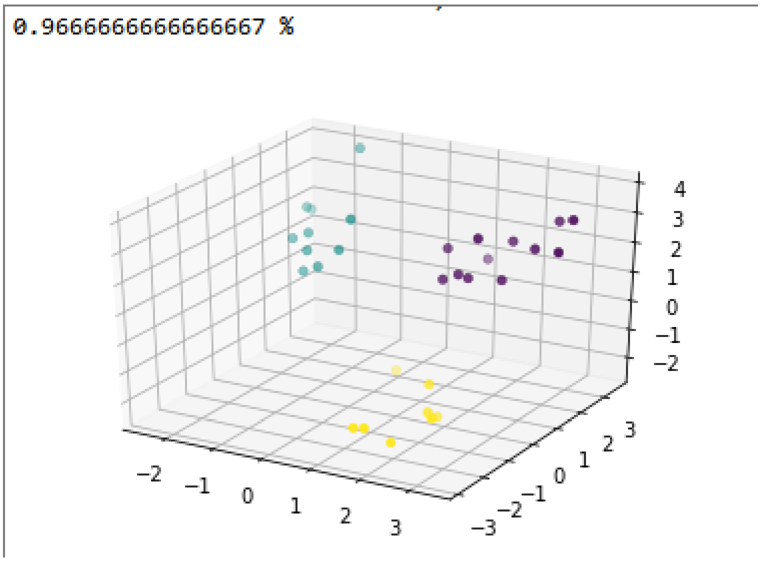
2. [10 POIN] Akurasi data uji



```python
# =============================================================
# Find the most optimal for G
# =============================================================
def searchOptimumG():
    index = 0
    Result = []
    while (np.floor(index) != 100):
        zzResults = mainAllTest(Data_train, Data_test, index)
        Result.append([validationTest(zzResults), index])

        index += .1

    df = pd.DataFrame(Result)
    df.to_csv('result5.csv', header=None, index=False)

#   searchOptimumG()
```



```python
def validationTest(resultTest):
    count = 0
    countDataTest = len(resultTest)
    for i, val in enumerate(resultTest):
        if (val[3] == val[5]):
            count += 1
    return count/countDataTest
```



```
0.9666666666666667 %
```